PARETO-OPTIMAL SITUATON ANALYSIS FOR SELECTION OF SECURITY MEASURES

Andres Ojamaa and Enn Tyugu
Institute of Cybernetics of
Tallinn University of Technology
Tallinn, Estonia

Jyri Kivimaa
Estonian Defence Forces Training and Development
Centre of Communication and Information Systems
Tallinn, Estonia

ABSTRACT

A methodology of selection of security measures is presented and a prototype implementation in the form of a hybrid expert system is described. This expert system is applicable, first of all, in the security management. It enables a user to select security measures in a rational way based on the Pareto optimality computation using a discrete dynamic programming method. This enables one to select rational countermeasures taking into account the available resources instead of using only hard constraints prescribed by standards. The prototype expert system is presented that provides a rapid security solution for a class of known information systems. Coarse-grained security can be analyzed in such a way at present, using a finite number of levels (security classes) as security metrics. This is a basis of the graded security methodology.

1. INTRODUCTION

Selection of security measures is a complex problem due to the fact that multiple objectives must be achieved at the same time. Considering data security, the security goals can be confidentiality, integrity and availability. Besides that, a security officer may want to keep costs reasonably low from one side, and reach the security goals with as high confidence as possible. Low cost and high confidence are two universal goals. The complexity has been an obstacle to finding optimal solutions for the security management problem. Another obstacle has been the absence of reliable metrics for measuring the said goals[1].

[1]"Good metrics are those that are SMART, i.e. specific, measurable, attainable, repeatable, and time-dependent, according to George Jelen of the International Systems Security Engineering Association [1]. Truly useful metrics indicate the degree to which security goals, such as data confidentiality, are being met, and they drive actions taken to improve an organization's overall security program [2]."

978-1-4244-2677-5/08/$25.00 © 2008 IEEE

Graded approach has been applied earlier in standards covering areas other than information security [3]. In recent years a graded security method has been developed and used in a number of areas, not necessarily in information assurance [4]. This method relies on a coarse-grained metrics for the security goals and achieved confidences. It is successfully applied as a basis for security standards that prescribe concrete security measures for achieving a required level of confidence for each security goal [5, 6]. The method is not immediately applicable for finding an optimal solution of the security problem.

We are going here to use the metrics of the graded security method and build a model that binds taken security measures with costs and confidences of achieving the goals. We introduce a fitness function that presents by one numeric value the integral confidence of achieving the security goals. This allows us to formulate a problem of selecting security measures as an optimization problem in precise terms. However, we still have two goals: to minimize the costs and to maximize the integral security confidence. This problem will be solved by means of building a Pareto optimality tradeoff curve that explicitly shows the relation between used resources and security confidence. Then, knowing the available resources, one can find the best possible security level that can be achieved with the resources and find the security measures to be taken. From the other side – if the required security level is given one can find the resources needed and the measures that have to be taken. This requires solving an optimization problem for each value of resources. As the number of possible security measures (that are in principle the independent variables of the optimization problem) is large, we have grouped the measures into security measures groups that will be characterized by security confidence levels. Taking the confidence levels of the groups as independent variables, we get an optimization problem of a reasonable size that can be solved by means of a discrete dynamic programming method.

The presented method of finding optimal security measures is in principle applicable in different situations, in particular, for designing overall security of a communication network, for designing a security of a critical information infrastructure of a bank etc. However, the method requires considerable amount of data that bind costs and confidences with security measures groups as well as expert knowledge that binds concrete security measures with a selected security confidence requirements level of a group. In the end of the present paper we give an example of an expert system developed for banking security that has the data and has been used for experimenting. Most of the expert knowledge of this kind can be extracted from standards or internal security policies of the bank or other organization that must have them before trying to optimize the security.

2. GRADED SECURITY MODEL

In the present section we briefly explain the basic concepts of the graded security model that gives functional dependencies for our optimization method. We are going to use integrated security metrics for representing the overall security of a system. Conventional goals of security are *confidentiality (C), integrity (I), availability (A),* and *non-repudiation (N)*. The model can be extended by including additional security goals. A finite number of security levels are introduced for each goal. This is a coarse-grained metrics, but the only available in this context at present. We use four levels 0, 1, 2, 3 for representing required security, but the number of levels can vary for different measures. The lowest level 0 denotes absence of special protective measures. *Security class* of a system is determined by security requirements that have to be satisfied. It is determined by assigning levels to goals, and is denoted by a respective tuple of pairs, e.g. `C2I1A1N2` for the system that has second level of confidentiality `C`, first level of integrity `I` and availability `A` and second level of non-repudiation `N`.

To achieve the security goals, proper *security measures* have to be taken. There may be a large number (hundreds) of measures. It is reasonable to group them into *security measures groups* g_1, g_2, $\ldots$, g_n. The grouping should be done in such a way that measures of one and the same group will be always used for achieving one and the same level of security. We will need a function f that produces a set of required security measures $f(l, g)$ for a given security measures group g and a security level l of the group.

A security class determines the required security level for each group of security measures. Let us denote by s a respective function that produces a security level $s(c, g)$ for a group g when the security class is c. *Abstract security profile* is an assignment of security levels (0, 1, 2 or 3) to each group of security measures. This can be expressed by the tuple $p = (s(c, g_1), s(c, g_2), \ldots, s(c, g_n))$, where p denotes the abstract security profile an the elements of the tuple p are indexed and appear in the tuple in the same order as the groups of security measures $g_1, g_2, \ldots, g_n$ have been indexed.

For n security measures groups we have totally 4^n abstract security profiles to be considered. The number of security measures groups may be in practice up to 20 or even more. This gives a number of abstract security profiles: 4^{20}. (If we had considered all security measures without grouping them, then we had got an incomprehensibly large number of security profiles – 4^k, where k is several hundreds.)

Knowing the cost function h that gives the costs $h(l, g)$ required for implementing security measures of a group g for a level l, one can calculate the costs of implementing a given abstract security profile:

$$costs(p) = \sum_{i=1}^{n} h(l_i, g_i), \text{ where } p = (l_1, l_2, \ldots, l_n)\,.$$

Our goal is to keep the value $costs(p)$ as low as possible.

The information for calculating values of functions f, h, c and s should be kept in the knowledge modules of an expert system of security measures.

It is assumed that, applying security measures, one achieves security goals with some confidence. The security confidence q of a group g that satisfies the security level l is given by a function $q(l, g)$ and it is a numeric value between 0 and 100 for each group of security measures.

We describe overall security of a system by means of an *integrated security metrics* that is a weighted mean security confidence S, called also *integral security confidence*:

$$S = \sum_{i=1}^{n} a_i q_i\,,$$

where q_i is security confidence of the i-th security measures group, a_i is a weight of the i-th group, and

$$\sum_{i=1}^{n} a_i = 1\,.$$

In the simplest case $a_i = 1/n$, and the integral security confidence is the average confidence of security measures groups. Also the information about the weights a_i, as well as about the function q must be presented in an expert system.

3. OPTIMIZATION TECHNIQUE

Now we can formulate an optimization problem as follows: "find the abstract security profile p with the best (highest) value of integral security confidence S for given amount of resources r, so that $costs(p) \leq r$." We have introduced all functions needed for calculating S and $costs$ in the previous section. Independent variables whose values have to be found by optimization are the security levels assigned to security measures groups: $l_1, l_2, \ldots, l_n$. If the security class c is given, then the solution has to satisfy also the constraints

$$l_i \geq s(c_i, g_i),\; i = 1, 2, \ldots, n\,.$$

Remark. The graded security model presented in Section 2 is usually used for finding (for a given security class) the required security levels of security measures groups and respective costs and concrete measures to be taken. This problem is considerably simpler that the optimization problem considered here.

Let us solve the optimization problem in the general case when also a security class is given. First, a security class prescribes only minimal security requirements and respectively – spending of some minimal amount of resources r_{min}. It is easy to calculate also resources r_{max} that can be reasonably spent for achieving the maximal possible integrated security level –

$$S_{max} = \sum_{i=1}^{n} a_i q_{max\,i}\ ,$$

where $q_{max\,i}$ is maximal security confidence of the i-th group of security measures.

Applying some resources between the values r_{min} and r_{max}, one can get better security in a rational way. Now we have an optimization problem with two goals: to minimize resources on the interval $[r_{min}, r_{max}]$ and to

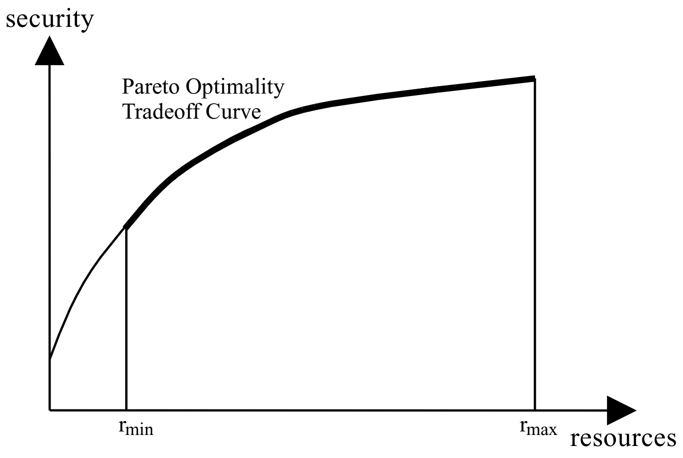


Figure 1. Search of optimal security along resource dimension

maximize security, guaranteeing at least the levels prescribed by a given security class. We reduce this problem to a simpler one that will be solved many times. We find an optimal security solution – the solution that has maximal value of S, for a fixed value of resources. We repeat this optimization for as many values of resources as needed. In this way we get a curve that shows the best possible value of fitness function S for every value of resources used, see Fig. 1. This curve is called a *Pareto optimality tradeoff curve* for resources and security. In the case when the minimal security requirements are not strict for security measures groups, it is reasonable to compute Pareto optimality even for resources less than r_{min}. This can be done, if the optimization procedure is sufficiently fast, like in our case.

The exhaustive search of optimal solutions for m possible values of resources, n security measures groups and k security levels requires testing (calculating weighted mean confidentiality) of mk^n points.

Building optimal solutions gradually, for $1, 2, \ldots, n$ security measures groups enables us to use discrete dynamic programming, and to reduce considerably the search. Indeed, the fitness function S defined on intervals from j to k as

$$S(j,k) = \sum_{i=j}^{k} a_i l_i\,,$$

is additive on the intervals, because from the definition of the function S we have

$$S(1,n) = S(1,k) + S(k,n)\,.$$

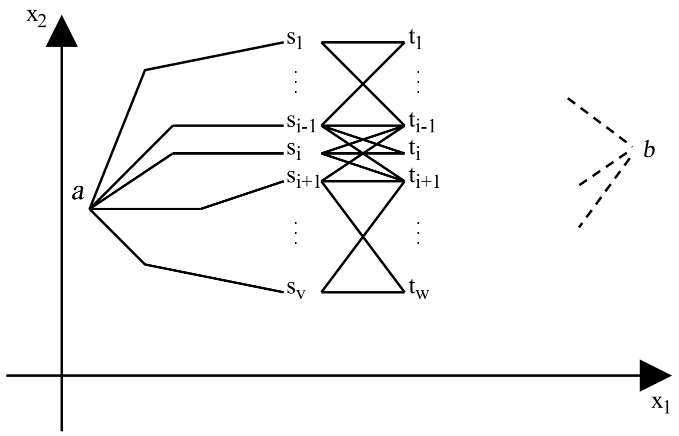


Figure 2. Resource assignment by means of discrete dynamic programming

This means that one can build an optimal resource assignment to security measures groups gradually, as a path in the space with coordinates x_1, x_2, where x_1 equals to the number of security measures groups that have got resource (i.e. $x_1 = k$) and x_2 equals to the amount of used units of resources $(1, 2, \ldots, 1000$ in our example). The discrete dynamic programming method requires using of a finite number of values of a resource (x_2). This number of values depends on the precision that is required. A precision that can be achieved using expert knowledge is not very high, usually a hundred points is sufficient. As our optimization procedure works sufficiently fast we are using 1000 points. Fig. 2 shows a search step, where known optimal partial solutions (assignments of resources to already tested security measures groups) are the paths from initial state a (where no resources are assigned) to intermediate states $s_1, \ldots, s_v$. The aim is to find one step longer optimal paths from a to the states $t_1, \ldots, t_w$ that follow the states $s_1, \ldots, s_v$. This can be done for each step by trying out all possible continuations of the given partial optimal paths to $s_1, \ldots, s_v$ as shown in Fig. 2. This algorithm requires testing of $m^2 n$ points (m is number of possible values of resources, n is number of security measures groups).

In Fig. 3 it is shown how the number of search steps (and consequently search time) depends on the number of security measures groups for the number of groups up to 10. Our method has linear complexity, the search time grows linearly with the number of groups. The time for exhaustive search grows exponentially.

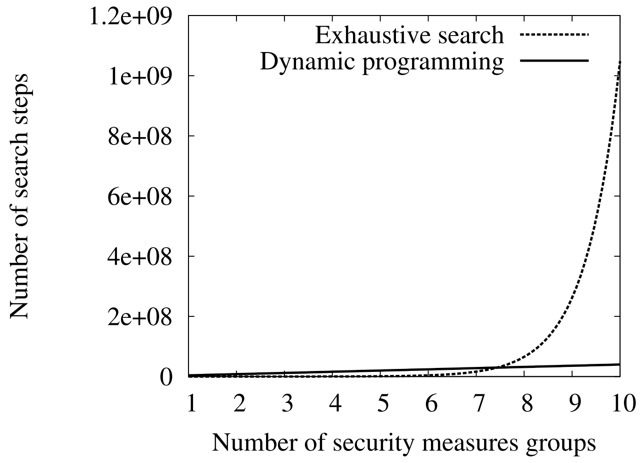


Figure 3. Complexity of search

4. APPLICATION EXAMPLE

We have developed a prototype of a security expert system for selecting security measures in banking. This expert system has been developed in a visual programming environment CoCoViLa [8]. Let us explain its functioning on an example. Here we use the following four security goals: confidentiality (C), integrity (I), availability (A) and satisfying mission criticality (M). We use the following nine security measures groups in our simplified example which are based on an educational information assurance video game CyberProtect [7]:

- firewall,
- access control,
- intrusion detection,
- encryption,
- user training,
- antivirus software,
- segmentation,
- redundancy,
- backup.

After selecting security levels for a security measures group, one can find a set of concrete measures to be taken. For example, in the case of the security level 1 for the group “User training” the following measures can be found from the expert system:

- New employees must be instructed for security – procedures and practice must be explained.
- An employee must know security related rights and obligations, must understand security practice, know about handling of passwords and keys.

- An employee must be instructed about security regulations and should be motivated to follow the regulations. Help about security must be available for all IS users.

The main window of the expert system is presented in Fig. 4. A menu bar in the upper part of the window provides buttons for building a visual specification of the problem to be solved. These are buttons for adding different kinds of components, and tools for manipulating components on the scheme and connecting components through ports. From left to right the buttons are:

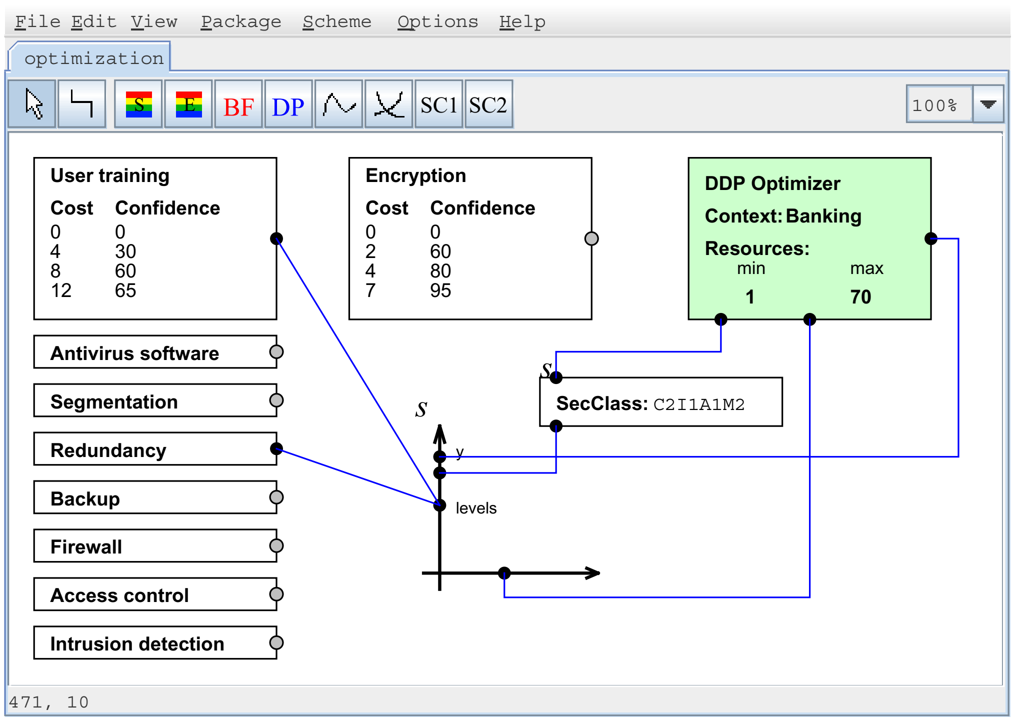
- selection tool,
- connection line,
- simple security measures group (uses default values defined by its name and current context),
- expanded security measures group with explicit values,
- brute-force optimizer,
- dynamic programming optimizer,
- simple graph,
- multiple graphs,
- security class evaluator,
- extended security class evaluator.

The window includes a visual specification of our problem. The specification is a scheme where components are security measures groups and other software components that are used for solving the problem. The scheme has been composed gradually by adding and connecting new components and editing properties of the components. In the given scheme we see images of all security measures groups. Besides the security measures groups there are three components *Optimizer*, *SecClass* and *GraphVisualizer* shown in the scheme. Two groups "User training" and "Encryption" are represented by expanded components that have specific values of cost and confidence related to security levels explicitly given as an input. Standard values of cost and confidence are used for other groups. They are given in the expert knowledge modules. We have to solve the problem in the context of banking and can use resources measured in some units on the interval from 1 to 70 that is shown in the *Optimizer* block. The security class `C2I1A1M2` is given as a separate block as well. Some blocks in the main window are connected through ports. This allows us to show which values of security should be visualized ("User training" and "Redundancy" in the present case) etc. The expected outcome is a graph produced by the *GraphVisualizer* that shows the weighted mean security confidence depending on the resources that are used in the best possible way. The graph should also indicate whether the security goals specified by the security class can be achieved with the given amount of resources. Besides that, the curves showing security confidence provided by user training and redundancy will be shown, see the respective connection lines between the visual images.

5. OPTIMIZATION RESULTS

In Fig. 5 there is a window showing the optimization results. The upper curve (Confidence) represents the optimal value of weighted mean security confidence depending on the resources that are used in the best possible way. This curve is further divided into four parts to visualize to which degree the optimal result satisfies the security requirements given by the security class. The first part (thin black line for the costs up to 27) indicates the interval of resources where none of the four (in our example) security goals can be achieved. The second part (thin grey line, three separate segments) shows that at least one of the security goals is satisfied while also at least one is not. The third part (thick black line) represents the amount of resources that, when used optimally, would result in satisfying the requirements exactly. One should note that this coincidence of the optimal security profile and the security requirements does not always exist. The last part of the graph (thin black line, again) shows the amounts of resources that are more than is strictly needed to satisfy the requirements. It is interesting to notice that on the interval of costs from 36 to 45 units it is possible to satisfy all security goals, because already spending 34 units enables one to do this. However, the solutions with highest values of the weighted mean security confidence do not satisfy all security goals on this interval.

The lower graphs indicate (on the scale shown on the right) the optimal levels of two measures groups corresponding to the given amount of resources. These graphs are not necessarily monotonic as can be seen in this example at the resource values 35 and 36. When there are 35 units of resources available it is reasonable to apply the measure "User training" at level 2. Having one more unit of resources, a better overall security confidence level is achieved by taking all resources away from "User training" and investing into the "Redundancy" measures group to achieve level 3.



User training

Cost	Confidence
0	0
4	30
8	60
12	65

Encryption

Cost	Confidence
0	0
2	60
4	80
7	95

DDP Optimizer
Context: Banking
Resources:

min	max
1	70

SecClass: C2I1A1M2

Figure 4. Main window of security expert system

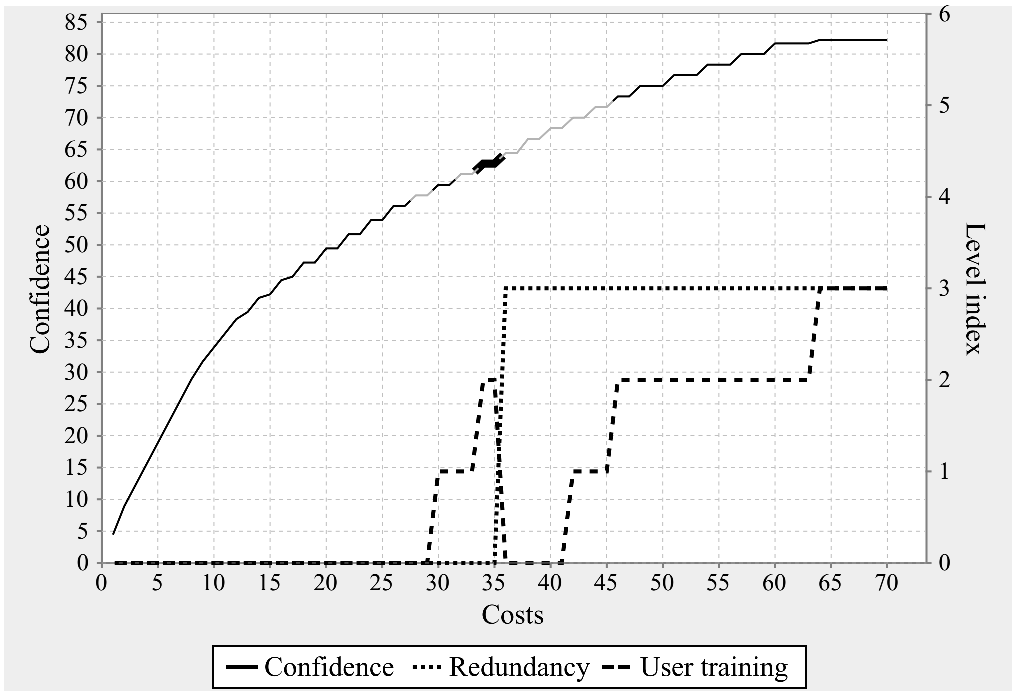


Figure 5. Solution of the problem

6. CONCLUDING REMARKS

In the present work we have developed a method for systematic design of a security solution of an information or communication system, and the method is explained on an example from the banking security. The method relies on a graded security model used in practice in different applications. The novelty of the method is, first, the usage of an advanced optimization technique based on discrete dynamic programming and, second, the output of many alternative solutions in the form of a Pareto optimality tradeoff curve that enables the user to select the best security solution depending on availability of resources.

Another novelty is introduction and usage of an integral security measure in the form of a weighted mean security confidence. The method performs security situation analysis using coarse-grained metrics for security levels of partial solutions (security measures groups) from one side, and an integrated security metrics in the form of weighted mean security confidence from the other side. A tool developed as a prototype supports visual presentation of a general view of a security situation and enables one to perform the situation analysis on different levels of details, e.g. using standard functions of confidences and costs or presenting them as additional inputs. Time required for automated analysis, when a set of input data is given, is only a few seconds. This enables one to perform the analysis rapidly for many different assumptions.

We understand that wider application of this method will depend on the availability of expert knowledge that binds costs and security confidence values with taken security measures. This knowledge can be collected only gradually, and will depend on the type of the critical infrastructure that must be protected. However, our expectation is that more expert knowledge will be collected when interactive analysis applications with graphical user interface such as the prototype presented in this paper become available.

ACKNOWLEDGEMENTS

We thank the Estonian Ministry of Defence and the Estonian Defence Forces Training and Development Centre of Communication and Information Systems for the support of this work. The first author would like to thank the Estonian Information Technology Foundation and the Tiger University program for partial support of this work.

REFERENCES

[1] G. Jelen. *SSE-CMM Security Metrics*. NIST and CSSPAB Workshop, Washington, D.C., 13–14 June 2000.

[2] S. C. Payne. *A Guide to Security Metrics*. SANS Reading Room, 2006. http://www.sans.org/reading_room/whitepapers/ (10 Sep 2008)

[3] C. E. Pasterczyk. *A graded approach to ISO 9000 implementation for records managers*. Association of Records Managers and Administrators international annual conference, Toronto (Canada), 25–29 September 1994.

[4] Y. Kang, C. H. Jeong, D. I. Kim. *Regulatory approach on digital security of instrumentation, control and information systems in nuclear power plants*. Korea Institute of Nuclear Safety. Daejeon, Korea. http://entrac.iaea.org/I-and-C/TM_IDAHO_2006/CD/IAEA%20Day%202/Kang%20paper.pdf (10 Sep 2008)

[5] German Federal Office for Information Security (BSI). *IT Baseline Protection Manual*. 2005. http://www.bsi.de/gshb/ (10 Sep 2008)

[6] U. S. Department of Defense. *National Industrial Security Program Operating Manual (NISPOM)*. 2006.

[7] U. S. Department of Defense, Defense Information Systems Agency. *CyberProtect, version 1.1*. July 1999. http://iase.disa.mil/eta/product_description.pdf (10 Sep 2008)

[8] P. Grigorenko, A. Saabas, E. Tyugu. *Visual tool for generative programming*. ACM SIGSOFT Software Engineering Notes, 2005, 30, 5, 249–252.